Imperial College London

Lecture 13

Impulse Response & Filters

Peter Cheung
Dyson School of Design Engineering

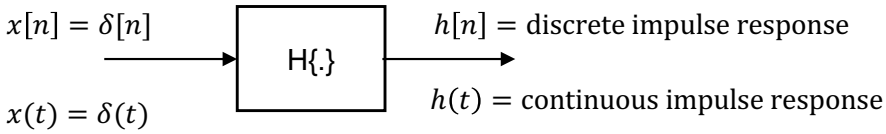
URL: www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/
E-mail: p.cheung@imperial.ac.uk

In this lecture, we will learn about:

1. Impulse response of a discrete system and what it means.
2. How impulse response can be used to determine the output of the system given its input.
3. The idea behind convolution.
4. How convolution can be applied to moving average filter and why it is called a Finite Impulse Response (FIR) filter.
5. Frequency spectrum of the moving average filter
6. The idea of recursive or Infinite Impulse Response (IIR) filter.

Impulse Response of a Discrete System

- Apply a discrete impulse *x[n]* to the input of a discrete time system, the output *y[n]* is known as the system's **impulse response *h[n]*.**



- The impulse response of a **linear** system defines and characterises the system completely – both its transient behaviour and its frequency response.
- This applies to both continuous time and discrete time linear systems.
- An impulse signal contains ALL frequency components (L3, S7). Therefore, applying an impulse to input stimulates the systems at ALL frequencies.
- Since integrating a unit impulse = a unit step function, we can obtain the step response of the system by integrating the impulse response.

If we apply an impulse at the input of a system, the output response is known as the system's **impulse response**: h(t) for continuous time, and h[n] of discrete time.

We have demonstrated in Lecture 3 that the Fourier transform of a unit impulse is a constant (of 1), meaning that an impulse contains ALL frequency components. Therefore apply an impulse to the input of a system is like applying ALL frequency signals to the system simultaneously.

Furthermore, since integrating a unit impulse gives us a unit step, we can integrate the impulse response of a system, and we can obtain it's step response.

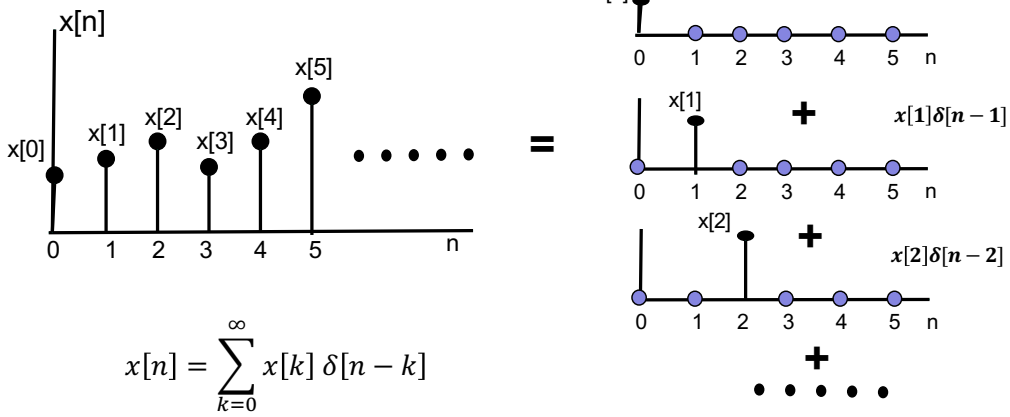
In other words, the impulse response of a system completely specify and characterise the response of the system.

So here are two important lessons:

1. Impulse response h(t) or h[n] characterizes a system in the time-domain.
2. Transfer function H(s) or H[z] characterizes a system in the frequency-domain.

Discrete signal as sum of weighted impulses

- Remember from L10, S7, we can represent a causal discrete signal *x[n]* in terms of sum of weighted delayed impulses:

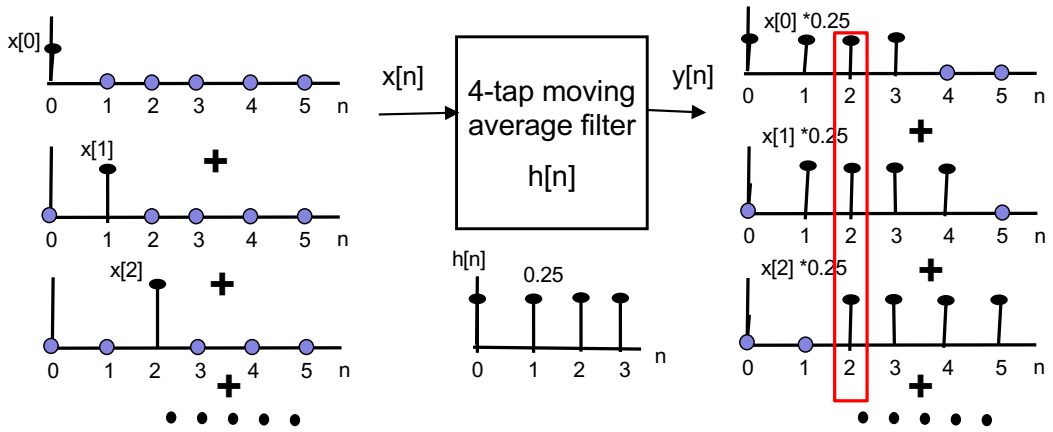


$$x[n] = \sum_{k=0}^{\infty} x[k]\,\delta[n-k]$$

Mathematically, it is really useful to model a discrete time signal x[n] consisting of sample sequence: {x[0], x[1], x[2] …} in terms of the unit impulse with different delays. We have already seen this in Lecture 11 before.

The plots on the right is the sequence x[n] decomposed into a sum of delay impulses at each sampling point, each being weighted by the signal x[n].

Impulse Response and Convolution



- We can therefore derive the output of a discrete linear system, by adding together the system's response to EACH input sample separately.
- This operation is known as **convolution**:

$$y[n] = x[n] * h[n] = \sum_{m=0}^{\infty} x[m]h[n-m]$$

Therefore when considering the response of a discrete system to any arbitrary input, it is useful to think of the inputs as a sum of lots of weighted impulses (delayed).

Let us consider the simple 4-tap moving average filter, whose impulse response h[n] is as shown.

x[0] sample will produce a response on the right side as shown. Similarly, we apply x[1], x[2] etc.

Now the output y[n] is just the sum of all the responses to each impulses.

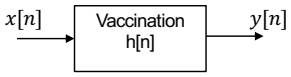
This operate of sum of product between x[.] and h[.], is represented mathmetically as (assume causality):

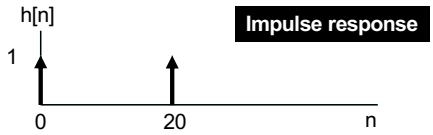
$$y[n] = x[n] * h[n] = \sum_{m=0}^{\infty} x[m]h[n-m]$$

This operation indicated by the '*' operator is known as convolution.

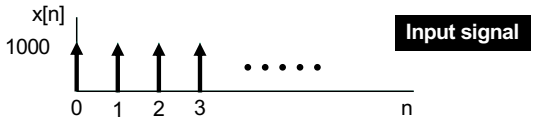
Convolution example - COVID vaccination

- Covid vaccine require two doses, 3 weeks apart. Impulse response h[n] for this vaccination system is:





- UK started vaccination of its elderly population with a plan of vaccinating, say, 1000 people per day after the first day (day 0). The input x[n] is:



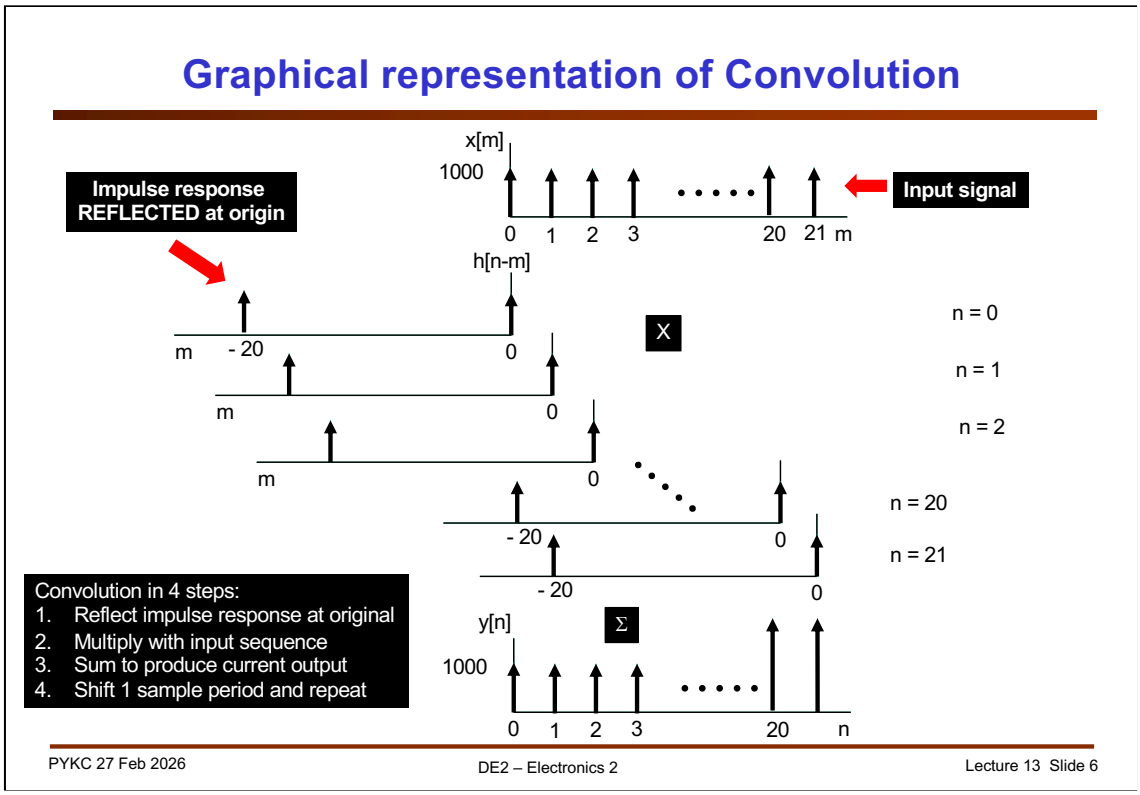
- How many doses would NHS need to provide from day 0? (i.e. y[n])

Let us consider convolution from a contemporary angle and ask: what problem does it solve?

COVID vaccine require two doses 3 weeks apart to be most effective. If we create a system to model this vaccination requirement, the impulse response of the system h[n] will be two impulses at day 0 and day 20.

If we now say that the government wants to start vaccinating 1000 people from day 0. This input x[n] can be module as a step function with a value of 1000.

Convolution answers the question: how many doses would one needs per day from day 0.



Computing y[n] is achieved by convolving x[n] with h[n], according to the following equations:

$$y[n] = x[n] * h[n] = \sum_{m=0}^{\infty} x[m]h[n-m]$$

Note that we change the variable for x[n] and h[n] in the summation, so that for every n in y[n], we do multiply and add.

This slide shows graphically what we are actually doing with this equation.

Let us first reflect h[n] on the y-axis (i.e. flip horizontally). Now we have the function h[n-m] for a given n.

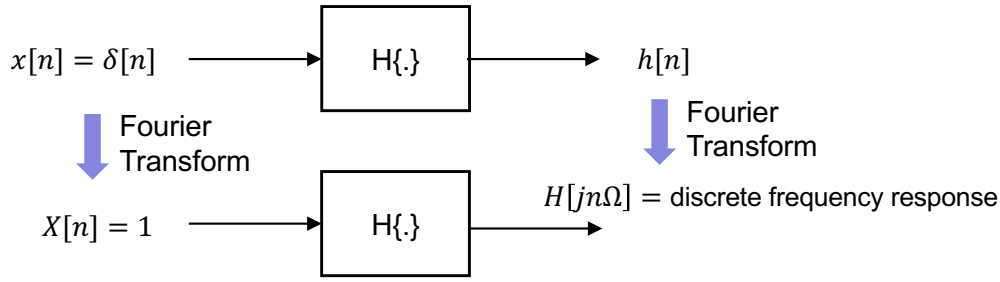
On day 0, we multiply h[0-m] with x[m] for all m, and add them together. This is the start of the vaccination, it is obvious that there are no other requirements but the 1000 dozes for the 1000 people. y[0] = x[0]*h[0] = 1000.

To work out requirement on day 1, we slide h[.] along 0 day and repeat the multiply and add. Each day, the answer is still 1000, until we get to day 20. Now people vaccinated on day 0 is now due a second dose. So:

y[20] = x[20]*h[0] + x[0]*h[-20] = 2000.

Impulse Response & Frequency Response

- Since a unit impulse contains all frequency, and its Fourier transform is a constant at 1 (see L3, S7), the Fourier transform of the impulse response h[n] or h(t) give us the systems' frequency response:

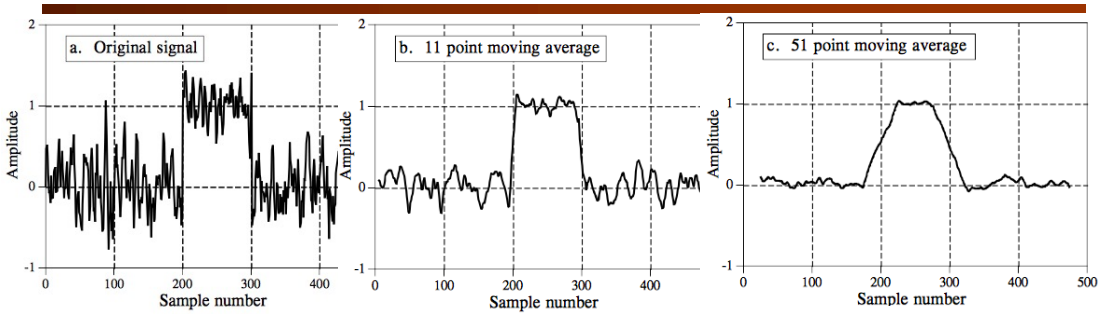


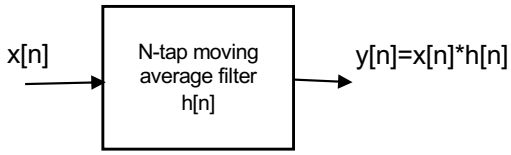
- This applies to both continuous time and discrete time linear systems.
- An impulse signal contains ALL frequency components (L3, S7). Therefore, applying an impulse to input stimulates the systems at ALL frequencies.

We have already seen that a unit impulse contains all frequency components. Therefore it is obvious that the impulse response is the system's response to ALL possible frequencies stimuli at the input.

This leads to the interesting result that the Fourier transform of the impulse response of a system give us the system's frequency response. This fact follows the argument that the Fourier transform is a linear transformation.

Moving Average Filter = FIR lowpass filter





- N-tap (or N point) moving average filter – higher N, lower the cut off frequency
- For a N-tap moving average filter, it impulse response has N impulses.
- If input x[n] has M non-zero samples (i.e. finite length), output y[n] is also finite in length, and has M+N non-zero samples. Hence the name Finite Impulse Response (FIR) filter.

Now let us consider some real discrete time system - the moving average filter.

We have seen in the last lecture and in Lab 6 that by averaging current input and N-1 previous input samples to produce current output has a low pass filtering effect (which is an averaging function). Here we show that a noisy signal being filtered by a 11-tap and 51-tap moving average filter.

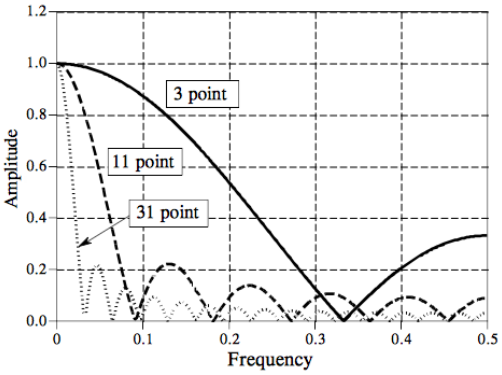
The procedure you followed in Lab 6 Task 1 is effectively performing the convolution operation: $y[n] = x[n]*h[n]$.

Frequency Response of N-tap moving average filter

- The impulse response of a moving average filter is a rectangular pulse.
- The Fourier transform of a rectangular pulse is of the form (sin x)/x or sinc(x) function (see Lecture 3 slide 6) in the case of continuous time.
- For discrete time case, the frequency response of a moving average filter with N taps (or points) is:

$$H[f] = \frac{\sin(\boldsymbol{f}N)}{N \sin f}$$

- Here *f* is normalised to 0 -> 0.5 x sampling frequency *fs*.



Remember that the frequency response of a system can be found by taking the Fourier transform of the impulse response.

The moving average filter has an impulse response = rectangular function rect(.).

From Lecture 3, slide 6, we have learned that the Fourier transform of a rectangular function is of the form of sin(x)/x, (or sinc(x)). Shown here is the frequency response of the moving average filter for different number of taps.

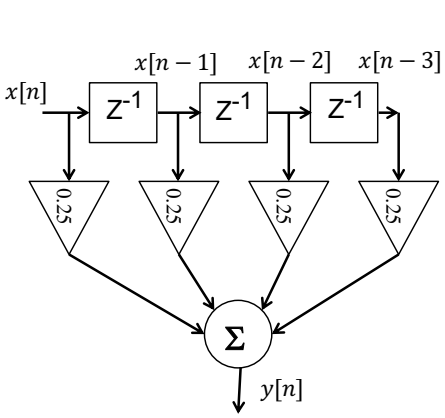
Note that the frequency axis is normalised (i.e. divided by) the sampling frequency fs. This is often the case in discrete time domain, we consider frequency in this way, as a ratio to the sampling frequency.

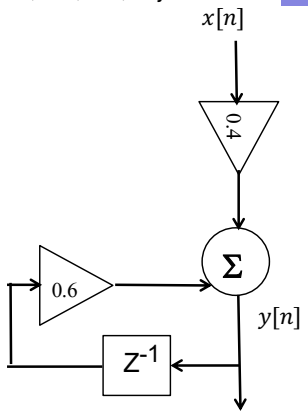
Recursive or Infinite Impulse Response Filter

FIR

x[n] =
{1.0, 1.0, 1.0, 1.0, 1.1, 0.8, 1.2, 0.9, 1.0, 1.2, 0.9, …}

IIR





y[n] = (0.25, 0.5, 0.75, 1.0, 1.025,
0.975, 1.025, 1.0, 1.0, 1.075, 1.0, …..}

y[n] = (0.4, 0.64, 0.784, 0.87, 0.962,
0.897, 1.018, 0.971, 0.983, 1.07,
1.002…..}

FIR filters are also known as **non-recursive filter** because output sample y[n] are ONLY depended on current input sample x[n], and previous input samples x[n-1], x[n-2] ….

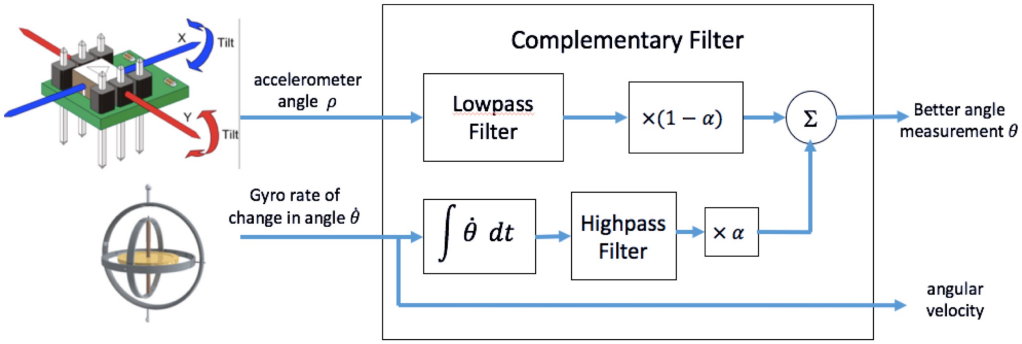
FIR filters require lots of multiplications and additions. Based on the convolution formula, a N-tap FIR filter in general would need N multiples and N adds.

A more computationally efficient approach would be to derive output samples y[n] from both current and past input samples, x[n], x[n-1], .. As well as previous outputs y[n-1], y[n-2] etc.

These type of filters are known as **recursive filters** or **Infinite Impulse Response** (IIR) filter.

It is interesting to consider the response of the FIR and the IIR filter to the input shown. In spite of a very simple structure (only 1 delay element, one multiply, and one add) of the recursive filter, it has an excellent low pass function as seen in the output sequence y[n].

Complementary Filter used with IMU



angle $\theta_{new} = \alpha \times (\ \theta_{old} + \ \dot{\theta}\ dt) + (1-\alpha) \times \rho$

where
- α = scaling factor chosen by users and is typically between 0.7 and 0.98
- ρ = accelerometer angle
- θ_{new} = new output angle
- θ_{old} = previous output angle
- $\dot{\theta}$ = gyroscope reading of the rate of change in angle
- dt = time interval between gyro readings

Remember that you implemented some kind of filter in Lab 3 to produce a much better angle measurement as shown here.

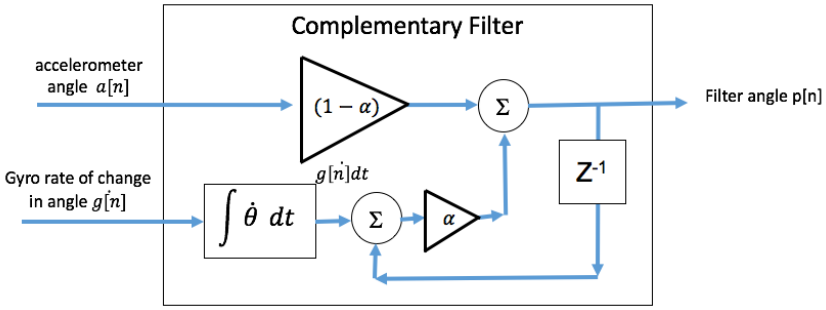
Now let us apply what we have learned to the Complementary filter used in the IMU measurements for the pitch and roll angles.

The block diagram is something that we have covered in Lab 3. The accelerometer angle is noisy, and we want to low pass filter (average) this to remove the noise due to movement.

The gyroscope measurement has drift (dc error, or slow changing value). We want to remove this via high pass filter.

We therefore derive the output pitch/roll angle, but by combining the readings from the accelerometer and the gyroscope.

Signal flow diagram model



$$p[n] = \alpha(p[n-1] + g[n]\times dt) + (1-\alpha)a[n]$$

```
def read_imu(dt):
    global g_pitch
    alpha = 0.7    # larger = longer time constant
    pitch = int(imu.pitch())
    roll = int(imu.roll())
    g_pitch = alpha*(g_pitch + imu.get_gy()*dt*0.001) + (1-alpha)*pitch
```

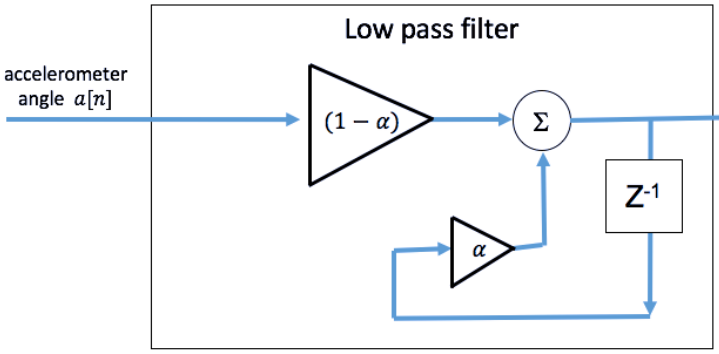

Before we consider how this filter works, let us cast the system in terms of difference equation. Shown here is the same set up as last slide, but now expressed as a difference equation.

The Micropython code to implement this is shown here. The equivalent version in Matlab for both pitch and roll angles is:

```
[p, r] = pb.get_accel();
[x, y, z] = pb.get_gyro();
dt = toc;
tic;
% integration for gyro angles
gx = max(min(gx+x*dt,pi/2),-pi/2);
gy = max(min(gy+y*dt,pi/2),-pi/2);

% complementary filtered angles
angle_x = alpha*(angle_x + x*dt) + beta*r;
angle_y = alpha*(angle_y + y*dt) + beta*p;
```

Lowpass filter the accelerometer data



$$p[n] = \alpha p[n-1] + (1-\alpha)a[n]$$

z-Transform

$$P[z] = \alpha z^{-1}P[z] + (1-\alpha)A[z]$$

$$P[z] - \alpha z^{-1}P[z] = (1-\alpha)A[z]$$

$$H[z] = P[z]/A[z] = \frac{1-\alpha}{1-\alpha z^{-1}}$$

- Now assume that gyroscope reading is zero (i.e. steady state tilt), g[n] = 0.
- Now the system is exactly the same as that in Lecture 12, slide 12.
- Therefore the accelerometer data a[n] is lowpass filtered!

Time Constant $\tau \approx \frac{\alpha}{1-\alpha} dt$

and dt is the sampling period

Let us for now assume that the gyroscope data is 0. The accelerometer data is applied to the discrete system as shown. Here we can immediate write down the difference equation as:

$$p[n] = \alpha p[n-1] + (1-\alpha)a[n]$$

This is exactly the same structure as the IIR filter we have seen two slides earlier (α=0.8).

Now if we take the z-transform of the difference equation, remembering that:

$$Z\{p[n-1]\} = P[z]z^{-1}$$

We get:

$$P[z] = \alpha z^{-1}P[z] + (1-\alpha)A[z]$$

$$H[z] = P[z]/A[z] = \frac{1-\alpha}{1-\alpha z^{-1}}$$

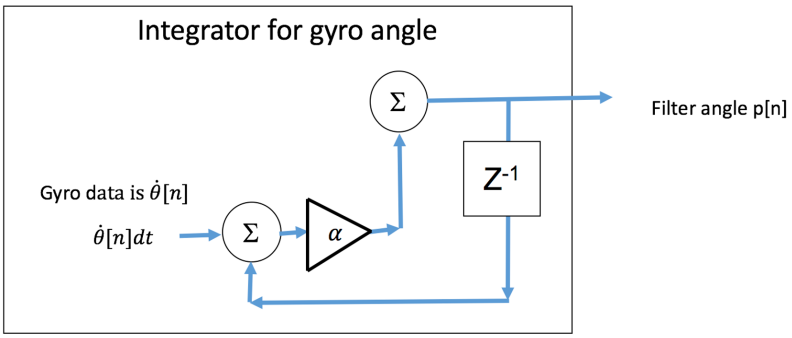
This is the discrete time transfer function of a typical low pass filter.

One important parameter of this filter is its **time constant** τ. It can be shown that if the sampling period (i.e. 1/fs) is dt, then

$$\text{Time Constant } \tau \approx \frac{\alpha}{1-\alpha} dt$$

Assuming that we have a sampling frequency of 100Hz, dt = 10ms and α = 0.8, the time constant is around 40msec. In other words, noise shorter than 40msec duration will be removed. If you want longer time constant, reduce value of α.

Integrating the gyroscope reading



$$p[n] = \alpha(p[n-1] + \theta[n]dt)$$

$$p[n] = \alpha^n p[0] + \alpha^n k$$

- Assume the gyro is not moving, but has a constant offset $\dot{\theta}_e$.
- dt is also constant.
- If $\alpha = 1$, $p[n]$ is a ramp with a gradient of $\dot{\theta}_e$.
- If $\alpha < 1$, then the effect of the error overtime diminishes to $\alpha^n \to 0$.

Now let us ignore the accelerometer signal, and only retain the gyroscope signal.

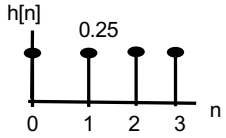
The integration process can be represented as a signal flow diagram and difference equation here. Assume that α = 1, then p[n] is the integral of the gyro data.

Question to ask:

1. Why is this prone to drift if the gyroscope has an offset error?
2. The drift error is a constant offset k, if α < 1, what will the effect of this error on p[n] over a long time?
3. Why is this equivalent to highpass filter?

Three Big Ideas (1)

1. A discrete time system can be characterized by its impulse response:
$$h[n] = b_0\delta[n] + b_1\delta[n-1] + b_2\delta[n-2] \ldots . + b_k\delta[n-k]$$



Impulse response of a 4-tap moving average filter

2. Once we know the impulse response $h[n]$, and the input sequence $x[n]$, we can find the output $y[n]$ by convolution:
$$y[n] = x[n] * h[n] = \sum_{m=0}^{\infty} x[m]h[n-m]$$

The three big ideas behind this lectures are:

1. A discrete time system can be characterized with its response to an impulse input (this is called the Impulse Response).

2. If you know the impulse response h[n], you can calculate the output sequence y[n] for any input sequence x[n] by performing the discrete convolution:

$$y[n] = x[n] * h[n]$$

Three Big Ideas (2)

3. Convolution operation can be performed in four steps:
   To obtain output y[n]:
   1) **Reflect** impulse response at n to get h[n-m]
   2) **Multiply** input sequence x[m] with h[n-m]
   3) **Sum** the product of the two sequences to get one output y[n]

$$y[n] = \sum_{m=0}^{\infty} x[m]h[n-m]$$

   4) **Advance** the reflected impulse response by one sample period and repeat to get the **next** y[n]

3. Graphically you can perform convolution by:

   a) Reflect the impulse response about the origin.
   b) Multiply element-by-element the input sequence and the impulse response.
   c) Add all products together to find the current output sample.
   d) Advance the reflected impulse response h[n-m] by one sample and repeat a) to c).